UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 10/607,591 | 06/27/2003 | Mark Ronald Plesko | 3382-64706 | 5996 |

26119 7590 07/14/2009

KLARQUIST SPARKMAN LLP
121 S.W. SALMON STREET
SUITE 1600
PORTLAND, OR 97204

| EXAMINER | |
|---|---|
| DAO, THUY CHAN | |

| ART UNIT | PAPER NUMBER |
|---|---|
| 2192 | |

| MAIL DATE | DELIVERY MODE |
|---|---|
| 07/14/2009 | PAPER |

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

| Office Action Summary | Application No. | Applicant(s) | |
|---|---|---|---|
| | 10/607,591 | PLESKO ET AL. | |
| | Examiner | Art Unit | |
| | THUY DAO | 2192 | |

*-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --*

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

1)☒ Responsive to communication(s) filed on *11 May 2009*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

4)☒ Claim(s) *1,3-7,9,10,12,14,15,17-21,24 and 26-37* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,3-7,9,10,12,14,15,17-21,24 and 26-37* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

**Application Papers**

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *27 June 2003* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *05/28/09*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

## DETAILED ACTION

1. This action is responsive to the amendment filed on May 11, 2009.

2. Claims 1, 3-7, 9, 10, 12, 14, 15, 17-21, 24, and 26-37 have been examined.

## Response to Amendments

3. In the instant amendment, claims 1, 12, and 14 have been amended; claims 13, 22, and 23 have been canceled; and claims 33-37 have been added.

## Response to Arguments

4. Applicants' arguments have been considered.

a) Claim 1 (Remarks, pp. 8-10):

The examiner notes that Applicants' amendments necessitated the new ground(s) of rejection presented in this Office action as set forth in details below.

b) Claim 6 (Remarks, pp. 10-12):

Claim limitations at issue "*determining whether to retain type information for one or more elements of the [intermediate language] representation*" (claim 6, lines 5-6).

The examiner respectfully disagrees with Applicants' assertions. Gordon explicitly teaches *determining whether to retain type information for one or more elements of the [intermediate language] representation* (e.g., FIG. 23, Intermediate Language Compilers for VC/VC++, Visual Basic, SQL;

FIG. 24, data types supported by a particular IL compiler, i.e., determining to retain type information of supported data types;

col.8: 3-6, "For non-primitive variables, however, complete type information must be maintained" (emphasis added); and

col.9: 58 – col.10: 23, maintaining type information in the lower 6 bits of flags.

Claim limitations at issue "*an unknown type, wherein the unknown type indicates that an element of the representation is of a type that is not known*" (claim 6, lines 7-8).

In an analogous art, Lidin further discloses rules for type-checking (e.g., chapter 7, Table 7-6, checking Native Types Defined in the Runtime) and a native type named "IUNKNOWN" (e.g., chapter 7, page 10).

Neither Gordon nor Lidin explicitly discloses said native type IUNKNOWN as a type *designated as an unknown type, wherein the unknown type indicates that an element of the representation is of a type that is not known.*

However, in an analogous art, Leach further discloses IUNKNOWN as *a type designated as an unknown type, wherein the unknown type indicates that an element of the representation is of a type that is not known* (e.g., col.6: 59 – col.7: 18, "IUNKNOWN" is the unknown type indicating that an element (interface) of IUNKNOWN is unknown).

c) Claim 12 (Remarks, pp. 12-13):

The examiner notes that Applicants' amendments necessitated the new ground(s) of rejection presented in this Office action as set forth in details below.

d) Claim 14 (Remarks, pp. 14-15):

The examiner notes that Applicants' amendments necessitated the new ground(s) of rejection presented in this Office action as set forth in details below.

e) Claim 24 (Remarks, page 15):

The Applicants asserted, "*Leach does not teach or suggest representing types in an intermediate language including an unknown type*" (page 15).

The examiner respectfully disagrees. As set forth in the previous Office action mailed February 9, 2009, pages 12-13, Lidin further discloses *rules for type-checking* (e.g., chapter 7, Table 7-6, checking Native Types Defined in the Runtime) and a native type named "IUNKNOWN" (e.g., chapter 7, page 10).

Neither Gordon nor Lidin explicitly discloses said native type IUNKNOWN as a type *designated as an unknown type, wherein the unknown type indicates that an element of the representation is of a type that is not known.*

However, in an analogous art, Leach further discloses IUNKNOWN as *a type designated as an unknown type, wherein the unknown type indicates that an element of the representation is of a type that is not known* (e.g., col.6: 59 – col.7: 18,

> "At run time, the word processing program would need to determine whether a spreadsheet object to be embedded supports the IDatabase interface. To make this determination, another interface is defined (that every spreadsheet object implements) with a function member that indicates which interfaces are implemented for the object. This interface is named IUnknown (and referred to as the unknown interface or the object management interface) and is defined as follows." (i.e., "IUNKNOWN" is the unknown type indicating that an element (interface) of IUNKNOWN is unknown).

f) Dependent Claims 3-5, 7, 9, 10, 15, 17-21, and 26-32 (Remarks, pp. 15-16):

Dependent Claims 3-5, 7, 9, 10, 15, 17-21, and 26-32 are also rejected based on virtue of their dependencies on the rejected base claims 1, 6, 14, and 24.

**Claim Rejections – 35 USC §103**

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. Claims 1, 3-5, 12, 29, 30, and 33-35 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No. 2002/0169999 A1 to Bhansali et al. (art made of

record, hereafter "Bhansali") in view of US Patent No. 2003/0101438 A1 to Mishra et al. (art made of record, hereafter "Mishra").

**Claim 1:**

Bhansali discloses *a method of type-checking a code segment written in a programming language comprising:*

*translating the code segment from the programming language to one or more representations of an a typed intermediate language* (e.g., [0010] and [0145]),

*wherein the one or more representations of the typed intermediate language are capable of representing programs written in a plurality of different source languages* (e.g., [0010] and [0052], Java, VB, C++), *and*

*wherein the one or more representations comprise a first object having a known type* (e.g., [0025] and [0116], types of parameters, local variables) *and wherein the translating comprises:*

*determining that the first object will be type-checked as a known type* (e.g., [0016], type checking and error handling, [0021], type checking of a variable), *and*

*based on the determining, designating the first object as having the known type* (e.g., [0062], known types such as real number, integer (array index), string (variable names)); *and*

*type-checking the one or more representations based on a rule set, wherein the rule set comprises rules for type-checking the type designated as the known type, wherein the known type indicates that an element of the representation is of a type that is known* (e.g., [0177], verification rules; [0182], type rules).

Bhansali does not explicitly disclose other limitations. However, in an analogous art, Mishra further discloses:

*determining that the first object will be type-checked as an unknown type, and based on the determining, designating the first object as having the unknown type* (e.g., [0059], unknown class name/field name/method/array size/array component type); *and*

*type-checking the one or more representations based on a rule set, wherein the rule set comprises rules for type-checking the type designated as the unknown type, wherein the unknown type indicates that an element of the representation is of a type that is not known* (e.g., [0059], type setting/checking is deferred until runtime).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine Mishra's teaching into Bhansali's teaching. One would have been motivated to do so to drop class/method/array information until runtime as suggested by Mishra (e.g., [0059]).

**Claim 3:**

The rejection of claim 1 is incorporated. Bhansali discloses *the rule set is selected from a plurality of rule sets* (e.g., [0182]-[0183]).

**Claim 4:**

The rejection of claim 3 is incorporated. Mishra discloses *only a fraction of the plurality of rule sets contain rules for type-checking a type designated as an unknown type, wherein the unknown type indicates that an element of the representation is of a type that is not known* (e.g., [0059]).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine Mishra's teaching into Bhansali's teaching. One would have been motivated to do so as set forth above.

**Claim 5:**

The rejection of claim 1 is incorporated. Bhansali discloses *the rule set further comprises rules for type-checking types representing categories of types found in a plurality of programming languages* (e.g., [0010] and [0052]).

**Claim 12:**

Bhansali discloses *a method of translating types associated with a plurality of programming languages to types of an intermediate language, the method comprising:*

*replacing the types associated with the plurality of programming languages with the types of the intermediate language* (e.g., [0016], type checking and error handling, [0021], type checking of a variable)*,*

*wherein the types of the intermediate language comprise plural programming language specific primitive types associated with the plurality of programming languages* (e.g., [0010] and [0052], Java, VB, C++) *and*

*a type designated as a known type, wherein the type designated as the known type has size information associated with it* (e.g., [0025] and [0116], types of parameters, local variables)*,*

*wherein the size information comprises size information of a machine representation of the type designated as the known type* (e.g., [0062], known types such as real number, integer (array index), string (variable names))*.*

Bhansali does not explicitly disclose other limitations. However, in an analogous art, Mishra further discloses:

*determining that the first object will be type-checked as an unknown type, and based on the determining, designating the first object as having the unknown type* (e.g., [0059], unknown class name/field name/method/array size/array component type)*; and*

*type-checking the one or more representations based on a rule set, wherein the rule set comprises rules for type-checking the type designated as the unknown type, wherein the unknown type indicates that an element of the representation is of a type that is not known* (e.g., [0059], type setting/checking is deferred until runtime)*.*

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine Mishra's teaching into Bhansali's teaching. One would have been motivated to do so to drop class/method/array information until runtime as suggested by Mishra (e.g., [0059]).

**Claim 29:**

Claim 29 is a computer-readable storage medium version, which recite(s) the same limitations as those of claim 1, wherein all claimed limitations have been addressed and/or set forth above. Therefore, as the reference teaches all of the limitations of the above claim(s), it also teaches all of the limitations of claim 29.

**Claim 30:**

The rejection of claim 1 is incorporated. Mishra discloses *the rule set further comprises rules for dropping type information for one or more elements of the representation by changing a known type of the one or more elements to the type designated as the unknown type* (e.g., ).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine Mishra's teaching into Bhansali's teaching. One would have been motivated to do so as set forth above.

**Claim 33 (New):**

Bhansali discloses *the method of claim 1, wherein the plurality of different source languages comprise at least one typed language and at least one assembly language* (e.g., [0052]-[0053], Java, C++, assembly language code).

**Claim 34 (New):**

Mishra discloses *the method of claim 1, wherein the rule set comprises a rule that causes the type-checking to not type-check the first object* (e.g., [0059]).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine Mishra's teaching into Bhansali's teaching. One would have been motivated to do so to drop class/method/array information until runtime as suggested by Mishra (e.g., [0059]).

**Claim 35 (New):**

Mishra discloses *the method of claim 1, wherein the one or more representations further comprise a second object having the first object's known type, and further comprising: determining that the second object will not be type-checked as an unknown type* (e.g., [0059])*, and*

*wherein the rule set comprises rules for type-checking types not designated as the unknown type, and wherein the type-checking comprises type-checking the second object using rules the rules for type-checking types not designated as the unknown type* (e.g., [0049] and [0056])*.*

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine Mishra's teaching into Bhansali's teaching. One would have been motivated to do so to drop class/method/array information until runtime as suggested by Mishra (e.g., [0059]).

7. Claims 6, 7, 9, 10, 24, 26-28, 31, 36, and 37 are rejected under 35 U.S.C. 103(a) as being unpatentable over Gordon (art of record, US Patent No. 6,560,774) in view of Lidin (art of record, "Inside Microsoft .NET IL Assembler") and Leach (art of record, US Patent No. 6,412,020).

**Claim 6:**

Gordon discloses *a method of selectively retaining type information during compilation in a code segment written in a programming language, the method comprising:*

*translating the code segment from the programming language to one or more representations of an intermediate language* (e.g., FIG. 2, col.6: 8-57)*;*

*for each representation, determining whether to retain type information for one or more elements of the representation* (e.g., FIG. 23, col.27: 4-33)*;*

*based on the determination, associating one or more elements of the representation with a type* (e.g., col.1: 54-62; col.6: 58 – col.7: 16; col.26: 5-14)*,*

*designated as an known type, indicating the element can be of any type; and type-checking the one or more representations based on a rule set* (e.g., FIG. 24, col.27: 35 – col.28: 29; col.23: 46 – col.24: 49),

*wherein the rule set comprises rules for type-checking the type designated as the known type* (e.g., col.17: 52 – col.18: 28; col.20: 44-57; col.27: 35 – col.28: 29).

In an analogous art, Lidin further discloses rules for type-checking (e.g., chapter 7, Table 7-6, Native Types Defined in the Runtime) and a native type named "IUNKNOWN" (e.g., chapter 7, page 10).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine Lidin's teaching into Gordon's teaching. One would have been motivated to do so to denote native types and perform type-checking in .NET framework in the runtime as suggested by Lidin (e.g., pp. 9-10 and 12-13).

Neither Gordon nor Lidin explicitly discloses IUNKNOWN as a type *designated as an unknown type, wherein the unknown type indicates that an element of the representation is of a type that is not known.*

However, in an analogous art, Leach further discloses IUNKNOWN as *a type designated as an unknown type, wherein the unknown type indicates that an element of the representation is of a type that is not known* (e.g., col.6: 59 – col.7: 18, "IUNKNOWN" is the unknown type indicating that an element (interface) of IUNKNOWN is unknown).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine Leach's teaching into Gordon and Lidin's teaching. One would have been motivated to do so to perform type-checking/determine whether an object to be embedded supports a specific interface as suggested by Leach (e.g., col.6: 59-67).

**Claim 7:**

The rejection of claim 6 is incorporated. Gordon discloses *the determination is based on a current stage of compilation, a characteristic of each representation, or the programming language* (e.g., col.6: 45 – col.7: 34; col.15: 37-67).

**Claim 9:**

The rejection of claim 6 is incorporated. Leach discloses *the type, designated as the unknown type, indicating the element can be of any type has size information associated with it* (e.g., e.g., col.6: 59 – col.7: 18).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine Leach's teaching into Gordon and Lidin's teaching. One would have been motivated to do so to perform type-checking/determine whether an object to be embedded supports a specific interface as suggested by Leach (e.g., col.6: 59-67).

**Claim 10:**

The rejection of claim 9 is incorporated. Leach discloses *generating code from at least elements associated with the type, designated as the unknown type, indicating the element can be of any type based on the size information* (e.g., col.6: 59 – col.7: 18).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine Leach's teaching into Gordon and Lidin's teaching. One would have been motivated to do so as set forth above.

**Claim 24:**

Gordon discloses *a method of representing types in an intermediate language comprising:*

*defining a plurality of types to be associated with elements of the intermediate language* (e.g., FIG. 2, col.6: 8-57),

*wherein one of the plurality of types indicates that an element of the intermediate language is associated with a type designated as an known type* (e.g., FIG. 23, col.27: 4-33; col.17: 52 – col.18: 28; col.20: 44-57; col.27: 35 – col.28: 29);

*wherein the type indicating that an element of the intermediate language is associated with the type designated as the known type has a size associated with it* (e.g., col.1: 54-62; col.6: 58 – col.7: 16; col.26: 5-14),

*wherein the size represents size of a machine representation of the type designated as the known type* (e.g., FIG. 24, col.27: 35 – col.28: 29; col.23: 46 – col.24: 49).

In an analogous art, Lidin further discloses rules for type-checking (e.g., chapter 7, Table 7-6, Native Types Defined in the Runtime) and a native type named "IUNKNOWN" (e.g., chapter 7, page 10).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine Lidin's teaching into Gordon's teaching. One would have been motivated to do so to denote native types and perform type-checking in .NET framework in the runtime as suggested by Lidin (e.g., pp. 9-10 and 12-13).

Neither Gordon nor Lidin explicitly discloses IUNKNOWN as a type *designated as an unknown type, wherein the unknown type indicates that an element of the representation is of a type that is not known.*

However, in an analogous art, Leach further discloses IUNKNOWN as *a type designated as an unknown type, wherein the unknown type indicates that an element of the representation is of a type that is not known* (e.g., col.6: 59 – col.7: 18, "IUNKNOWN" is the unknown type indicating that an element (interface) of IUNKNOWN is unknown).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine Leach's teaching into Gordon and Lidin's teaching. One would have been motivated to do so to perform type-checking/determine whether an object to be embedded supports a specific interface as suggested by Leach (e.g., col.6: 59-67).

**Claim 26:**

The rejection of claim 24 is incorporated. Leach discloses *an element of the intermediate language that was previously associated with another type is associated with the type indicating that the element is associated with the type designated as the unknown type* (e.g., col.6: 59 – col.7: 18).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine Leach's teaching into Gordon and Lidin's teaching. One would have been motivated to do so as set forth above.

**Claim 27:**

The rejection of claim 24 is incorporated. Gordon discloses *the plurality of types further comprises types representing categories of types found in a plurality of programming languages* (e.g., col.6: 8-57; col.15: 37-67).

**Claim 28:**

Claim 28 is a computer-readable storage medium version, which recite(s) the same limitations as those of claim 24, wherein all claimed limitations have been addressed and/or set forth above. Therefore, as the reference teaches all of the limitations of the above claim(s), it also teaches all of the limitations of claim 28.

**Claim 31:**

The rejection of claim 6 is incorporated. Leach discloses *at least one of the one or more representations of the intermediate language supports dropping type information by designating a type as an unknown type* (e.g., col.6: 59 – col.7: 18).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine Leach's teaching into Gordon and Lidin's teaching. One would have been motivated to do so as set forth above.

**Claim 36 (New):**

Leach discloses *the method of claim 6, wherein the translating further comprises:*

*identifying a virtual function call in the code segment; translating the virtual function call to an intermediate representation that when converted to computer-executable instructions* (e.g., col.7: 38 – col.8: 36) *and*

*executed on a computer cause the computer to perform a method including fetching a pointer associated with a virtual table, wherein the translating assigns a temporary variable to store the result of the fetching* (e.g., col.6: 59 – col.7: 18)*; and*

*wherein the determining further comprises not retaining type information for the temporary variable assigned to store the result of the fetching, because the type of the temporary variable is designated as the unknown type* (e.g., col.11: 18 - col.12: 9)*.*

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine Leach's teaching into Gordon and Lidin's teaching. One would have been motivated to do so as set forth above.

**Claim 37 (New):**

Leach discloses *the method of claim 6, wherein the type information for the one or more elements of the representation designated as an unknown type comprises a primitive type* (e.g., col.2: 1-56; col.6: 59 – col.7: 18)*.*

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine Leach's teaching into Gordon and Lidin's teaching. One would have been motivated to do so as set forth above.

8. Claims 14, 15, 17-21, and 32 are rejected under 35 U.S.C. 103(a) as being unpatentable over Gordon in view of Lidin, Leach, and Mishra.

**Claim 14:**

Gordon discloses *a system for type-checking an intermediate representation of source code in a compiler comprising:*

*one or more types associated with elements of the intermediate representation* (e.g., FIG. 2, col.6: 8-57)*,*

*wherein at least one of the types, designated as an unknown type, indicates an element can be of any type* (e.g., FIG. 23, col.27: 4-33; col.17: 52 – col.18: 28; col.20: 44-57; col.27: 35 – col.28: 29)*;*

*one or more rule sets comprising rules associated with the type* (e.g., col.1: 54-62; col.6: 58 – col.7: 16; col.26: 5-14),

*designated as the unknown type, indicating an element can be of any type* (e.g., FIG. 24, col.27: 35 – col.28: 29; col.23: 46 – col.24: 49);

*selectively retains type information for some elements of the intermediate representation and selectively does not retain type information for other elements of the intermediate representation* (e.g., col.17: 53 – col.18: 28; col.19: 60 – col.20: 10); *and*

*a type-checker for applying the one or more rule sets to the elements of the intermediate representation* (e.g., col.17: 52 – col.18: 28; col.20: 44-57; col.27: 35 – col.28: 29).

In an analogous art, Lidin further discloses rules for type-checking (e.g., chapter 7, Table 7-6, Native Types Defined in the Runtime) and a native type named "IUNKNOWN" (e.g., chapter 7, page 10).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine Lidin's teaching into Gordon's teaching. One would have been motivated to do so to denote native types and perform type-checking in .NET framework in the runtime as suggested by Lidin (e.g., pp. 9-10 and 12-13).

Neither Gordon nor Lidin explicitly discloses IUNKNOWN as a type *designated as an unknown type, wherein the unknown type indicates that an element of the representation is of a type that is not known.*

However, in an analogous art, Leach further discloses IUNKNOWN as *a type designated as an unknown type, wherein the unknown type indicates that an element of the representation is of a type that is not known* (e.g., col.6: 59 – col.7: 18, "IUNKNOWN" is the unknown type indicating that an element (interface) of IUNKNOWN is unknown).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine Leach's teaching into Gordon and Lidin's teaching. One would have been motivated to do so to perform type-checking/determine whether an object to be embedded supports a specific interface as suggested by Leach (e.g., col.6: 59-67).

Neither Gordon, Lidin, nor Leach explicitly discloses other limitations. However, in an analogous art, Mishra further discloses *selectively does not retain type information for an element of the intermediate representation by replacing a type associated with the element with the type, designated as the unknown type, indicating an element can be of any type* (e.g., [0059], unknown class name/field name/method/array size/array component type).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine Mishra's teaching into Gordon, Lidin, and Leach's teaching. One would have been motivated to do so to drop class/method/array information until runtime as suggested by Mishra (e.g., [0059]).

**Claim 15:**

The rejection of claim 14 is incorporated. Leach discloses *the type, designated as the unknown type, indicating the element can be of any type has size information associated with it* (e.g., col.6: 59 – col.7: 18).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine Leach's teaching into Gordon and Lidin's teaching. One would have been motivated to do so as set forth above.

**Claim 17:**

The rejection of claim 14 is incorporated. Gordon discloses *the one or more rule sets applied to the elements of the intermediate representation are selected based on the stage of compilation* (e.g., col.19: 60 – col.20: 10; col.23: 46 – col.24: 10).

**Claim 18:**

The rejection of claim 14 is incorporated. Gordon discloses *the one or more rule sets applied to the elements of the intermediate representation are selected based on a characteristic of the source code* (e.g., col.34: 41-64; col.35: 8-44).

**Claim 19:**

The rejection of claim 14 is incorporated. Gordon discloses *the one or more rule sets applied to the elements of the intermediate representation are selected based on the intermediate representation* (e.g., col.25: 51 – col.26: 25; col.35: 50 – col.36: 12).

**Claim 20:**

The rejection of claim 14 is incorporated. Leach discloses *only a fraction of the one or more rule sets contain rules for type-checking a type, designated as an unknown type, that indicates an element can be of any type* (e.g., e.g., col.6: 59 – col.7: 18).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine Leach's teaching into Gordon and Lidin's teaching. One would have been motivated to do so as set forth above.

**Claim 21:**

The rejection of claim 14 is incorporated. Gordon discloses *the one or more rule sets further comprise rules for type-checking types representing categories of types found in a plurality of programming languages* (e.g., col.27: 4-33).

**Claim 32:**

The rejection of claim 15 is incorporated. Leach discloses *the size information comprises size information of a machine representation of the type designated as the unknown type* (e.g., col.6: 59 – col.7: 18).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine Leach's teaching into Gordon and Lidin's teaching. One would have been motivated to do so as set forth above.

## Conclusion

9. Applicants' amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

10. Any inquiry concerning this communication should be directed to examiner Thuy Dao (Twee), whose telephone/fax numbers are (571) 272 8570 and (571) 273 8570, respectively. The examiner can normally be reached on every Tuesday, Thursday, and Friday from 6:00AM to 6:00PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam, can be reached at (571) 272 3695.

Any inquiry of a general nature of relating to the status of this application or proceeding should be directed to the TC 2100 Group receptionist whose telephone number is (571) 272 2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Twee Dao/
Examiner, Art Unit 2192

/Tuan Q. Dam/
Supervisory Patent Examiner, Art Unit 2192